Estadística básica en ciencias experimentales

Contents

Para una mejor visualización de este documento se puede visitar https://jabiologo.github.io/web/tutorials/estadisticaBasica.html

Presentación

En este espacio se encuentran los contenidos para las prácticas del tema **Estadística básica en ciencias experimentales** de la asignatura Bases de la investigación en ciencias experimentales, coordinada por Joaquín Vicente (Máster Universitario en Investigación Básica y Aplicada en Recursos Cinegéticos - MUIBARC). En los siguientes enlaces se pueden descargar los materiales se utilizarán durante el taller:

- Una **introducción al entorno R**, la herramienta que utilizaremos en las prácticas
- Una presentación de los contenidos prácticos.
- Scripts y [datos] necesarios para las sesiones prácticas (ver **Prácticas** más abajo).
- Datos necesarios los ejercicios (ver **Ejercicios** más abajo).

Este tema se ha realizado en base a los contenidos previamente desarrollados por Lorenzo Pérez Rodríguez y extendidos por Jéssica Jiménez-Peñuela, Daniel Parejo-Pulido y Javier Fernández-López.

Introducción al entorno R

¿Por qué R? R es un entorno y lenguaje de programación con un enfoque al análisis estadístico. Al preparar estas prácticas tuvimos serias dudas sobre si adoptar **R** como *herramienta* para aplicar los *contenidos* vistos en la teoría, u optar por un software con el clásico diseño "por ventanas". Entendemos que adoptar herramientas *complicadas* para trabajar contenidos que ya de por sí son complicados puede tener sus desventajas (estár más pendiente de aprender a utilizar la herramienta que a enterarse de la teoría, por ejemplo). También sabemos que R tiene una **curva de aprendizaje empinada**, sobre todo al principio. Sin embargo, hay varias razones para empezar a trabajar con esta herramienta:

- R es una herramienta libre y gratuita para todos los sistemas operativos.
- R no es solamente un software para análisis estadístico.
- Es muy probable que tarde o temprano tengáis que utilizar esta herramienta.

Descarga e instalación de R y RStudio Como ya hemos comentado, en esta práctica utilizaremos dos softwares que deberemos descargar e instalar en el caso de ser necesario:

- **R** lo podemos descargar en https://cran.r-project.org/ para los sistemas operativos Windows, macOS y Linux.
- **RStudio** lo podemos descargar en https://posit.co/download/rstudio-desktop/, también para cualquer sistema operativo

Una vez descargados, RStudio se vinculará con R automáticamente.

Un paseo por R y RStudio Es importante diferenciar entre R y RStudio. El software o lenguaje que utilizamos para los análisis estadisticos es **R**, por tanto podemos utilizar la consola de R por sí sola. **RStudio es un IDE** (del inglés *integrated development environment*), un entorno de desarrollo integrado, una "carcasa" que envuelve a R y facilita su uso. Es imporante familiarizarse con RStudio, ya que utilizaremos R a través de él. Tómate unos minutos para explorar sus diferentes paneles y personalizar su diseño:

- Diferencia entre *consola* y *script*
- Visualiza tu *entorno de trabajo*, tu ventana de *gráficos (plots)*, tus *librerías (paquetes)* cargadas, etc.
- También puedes cambiar el color de fondo, aumentar el tamaño de letra, etc.

Las `funciones()` Las **`funciones()`** son la "maquinaria" de R, las que realizan el trabajo. Se pueden identificar porque generalmente van seguidas de unos paréntesis entre los cuales se colocan sus **argumentos**. Los argumentos de una función son elementos que necesita esa función para ejecutarse y suelen ir separados por *comas* ,. Por ejemplo, existe una función que se llama "concatenar" (que en el lenguaje R se escribe `c()`), que simplemente sirve para unir en el mismo vector una serie de elementos (letras, números, etc.). Vamos a utilizar esa función para unir en un único vector una serie de números.

```
# La almohadilla se utiliza para incluir un comentario. Todo lo que se encuentre
# después de una almohadilla no se ejecutará en la consola.

# Unir los números 3, 7, 12 y 4 en un único vector
c(3, 7, 12, 4)
```

```
## [1]  3  7 12  4
```

- ¿Cuál es la función? ¿Cuáles son sus argumentos?

Imaginemos que ahora queremos hallar la media de esos números. Podemos utilizar otra función denominada `mean()` a la cual le introduciremos los numeros que hemos concatenado anteriormente como único argumento:

```
# Obtener la media de los números 3, 7, 12 y 4
mean(c(3, 7, 12, 4))
```

```
## [1] 6.5
```

Todas las funciones de R se alamcenan en "bibliotecas" o librerías (habitualmente denominados paquetes). Estos paquetes podemos entenderlos como cajas de herramientas donde se almacenan las herramientas que queremos utilizar. Hay algunos paquetes que vienen instalados y cargados por defecto en R. Sin embargo, otros los tenemos que descargar e instalar por primera vez y luego cargarlos en nuestra sesión cada vez que queramos utilizarlos. Por ejemplo, el paquete `lme4` se utiliza frecuentemente para ajustar modelos generales lineales mixtos. Podemos descargarlo, instalarlo y cargarlo en nuestra sesión de R de la siguiente manera:

```
# Descargamos e instalamos el paquete
install.packages("lme4")
# Cargamos el paquete en nuestra sesión
library(lme4)
```

Una de las grandes ventajas (o inconvenientes?) de R es que es un software libre, por lo que cualquiera puede desarrollar sus propios paquetes con las herramientas (funciones) que necesite y ponerlo a disposición de la

comunidad de usuarios. Si tenéis curiosidad, aquí podéis encontrar un pequeño tutorial sobre como hacerlo.

Los objetos Los **objetos** en R son los contenedores donde almacenamos los resultados (outputs) de las funciones. Podemos identificarlos porque suelen aparecer por primera vez precediendo a los caracteres `<-`, que simbolizan una flecha que señala hacia la izquierda. Cada vez que se quiera crear un objeto se le ha de dar un nombre, el que queramos, aunque suele ser conveniente darle un nombre que tenga sentido. Por ejemplo, vamos a almacenar en un objeto que vamos a llamar "numeros" la concatenación de valores que creamos anteriormente:

```
# Almacenamos en un objeto llamado "numeros" el resultado de concatenar 3, 7, 12 y 4
numeros <- c(3, 7, 12, 4)

# Ahora podemos "llamar" a "numeros" para ver qué tiene dentro
numeros
```

```
## [1]  3  7 12  4
```

```
# También podemos utilizar a "numeros" dentro de otra función, por ejemplo mean()
mean(numeros)
```

```
## [1] 6.5
```

```
# Por último podemos guardar dentro de otro objeto el resultado de la linea anterior

m <- mean(numeros)
m
```

```
## [1] 6.5
```

Todos los objetos en R tienen una clase, que informa sobre el tipo de objeto que es. Por ejemplo, si es un vector de números será `numeric`, pero si lo que almacena son caracteres su clase será `character`. Hay muchísimas clases de objetos (¡incluso se pueden crear clases nuevas!). Conocer la clase de nuestros objetos es muy important, puesto que **algunas funciones necesitan que sus argumentos sean de una clase específica**, y sino no funcionarán. Por ejemplo, no podemos hacer la media de las letras "a", "b" y "c", pero sí podremos hacer la media de los números 1, 2 y 3.

```
# Para averiguar la clase de un objeto usamos la función class()
class(numeros)
```

```
## [1] "numeric"
```

```
# Vamos a crear un objeto con los caracteres "a", "b" y "c"
letras <- c("a", "b", "c")

# Exploramos la clase de letras
class(letras)
```

```
## [1] "character"
```

```
# Vamos a intentar hacer la media de letras
mean(letras)
```

```
## Warning in mean.default(letras): argument is not numeric or logical: returning
## NA
```

```
## [1] NA
```

Entorno y directorio de trabajo Todos los objetos que vayamos creando o cargando en R se pueden visualizar en el panel `Environment` que suele situarse en la parte superior derecha en RStudio. En ese panel aparece información sobre los objetos, una previsualización o incluso, si el objeto lo permite, podemos

hacer click en él y visualizarlos de forma intuitiva en formato "hoja de cálculo". También es importante ser conscientes del directorio de trabajo en el que estamos trabajando, esto es, la carpeta en la que se guardarán los ficheros que salgan de R, o la carpeta desde donde se cargarán los ficheros. Para saber nuestro directorio de trabajo actual utilizamos la función `getwd()`, mienstras que para cambiarla utilizaremos `setwd("mi_directorio_de_trabajo_nuevo")`.

```
# Exploramos mi directorio de trabajo actual
getwd()
```

```
## [1] "/home/javifl/github/web/tutorials"
```

```
# Cambiamos el directorio de trabajo
setwd("/home/javifl/github/web/tutorials/estadisticaBasica_files")
```

Trabajo con `data.frame` Uno de los objetos que más vamos a utilizar en la práctica es el `data.frame`, que podría asimilarse a un fichero similar a lo que podríamos visualizar en una hoja de cálculo tipo Excel. Vamos a utilizar la función `read.csv` leer un archivo de texto plano delimitado por comas (CSV del inglés *"comma separated values"*) y ensayar algunas cosas con él. Podéis echar un vistazo a este archivo pinchando aquí.

```
# Guardamos el archivo en un obejto que denominaremos "datos"
datos <- read.csv("https://raw.githubusercontent.com/jabiologo/web/master/tutorials/estadisticaBasica_f

class(datos)
```

```
## [1] "data.frame"
```

```
# Podemos utilizar la función head() para explorar los primeros elementos de datos
head(datos)
```

```
##     peso    sex      hab      tri
## 1 419.35  Macho Agricola 47.87247
## 2 465.11 Hembra   Urbano 13.03228
## 3 434.17 Hembra   Urbano 18.05707
## 4 508.56  Macho Agricola 20.87707
## 5 514.75  Macho Agricola 15.29327
## 6 478.53  Macho   Urbano 15.52720
```

```
# Podemos utilizar la función str() para ver la estructura interna del data.frame
str(datos)
```

```
## 'data.frame':    300 obs. of  4 variables:
##  $ peso: num  419 465 434 509 515 ...
##  $ sex : chr  "Macho" "Hembra" "Hembra" "Macho" ...
##  $ hab : chr  "Agricola" "Urbano" "Urbano" "Agricola" ...
##  $ tri : num  47.9 13 18.1 20.9 15.3 ...
```

```
# O incluso podemos pedirle un resumen con summary()
summary(datos)
```

```
##       peso           sex                hab                 tri
##  Min.   :330.0   Length:300         Length:300         Min.   :-4.733
##  1st Qu.:430.9   Class :character   Class :character   1st Qu.:18.955
##  Median :454.9   Mode  :character   Mode  :character   Median :25.853
##  Mean   :453.0                                         Mean   :25.780
##  3rd Qu.:475.9                                         3rd Qu.:32.421
##  Max.   :555.1                                         Max.   :52.502
```

Tenemos varias formas de manejar los datos contenidos en un `data.frame`. Por ejemplo, si queremos explorar alguna columna podemos utilizar el símbolo del dólar `$` para seleccionarla por su nombre. Por ejemplo

`datos$peso`. Si lo que queremos es seleccionar un dato en concreto, podemos utilizar los corchetes `[]` para acceder a los elementos del `data.frame` indicando su posición en cuanto a filas y columnas separados por una coma con la forma `datos[fila,columna]`. Por ejemplo, si queremos seleccionar el dato que está en la fila 3 y columna 4, podemos emplear `datos[3,4]`. Si dejamos uno de los dos huecos en blanco, obtendremos toda la fila o toda la columna. Por ejemplo, si quremos todos los datos de la fila 7, podemos hacer `datos[7,]`. Veamos estos ejemplos.

```
# Visualizamos los nombres de las columnas de datos
colnames(datos)
```

```
## [1] "peso" "sex"  "hab"  "tri"
```

```
# Visualizamos los 15 primeros elementos de la columna peso
datos$peso[1:15]
```

```
##  [1] 419.35 465.11 434.17 508.56 514.75 478.53 429.03 463.79 445.34 412.27
## [11] 443.78 329.97 414.74 461.40 388.19
```

```
# Visualizamos el dato que está en la fila 3 y columna 4
datos[3,4]
```

```
## [1] 18.05707
```

```
# Visualizamos todos los datos de la fila 7
datos[7,]
```

```
##     peso    sex    hab      tri
## 7 429.03 Hembra Urbano 32.48139
```

Por último, vamos a ver una forma más moderna (y cómoda) de trabajar con los datos a través de `tydiverse`. Tydiverse es una coleccion de paquetes de R con una sintaxis en teoría más intuitiva que se centran en el manejo de datos. Utilizan el formato de "tuberías" o pipes mediante el símbolo `%>%`. Veamos un ejemplo.

```
# Instalamos (en caso necesario) y cargamos las librerías de tidyverse
# install.packages(tidiverse)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     1.4.0
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Tomamos el objeto datos, y lo filtramos dejando tan solo los machos
datosMacho <- datos %>% filter(sex == "Macho")

# Visualizamos los primeros elementos del nuevo objeto
head(datosMacho)
```

```
##     peso   sex      hab      tri
## 1 419.35 Macho Agricola 47.87247
## 2 508.56 Macho Agricola 20.87707
## 3 514.75 Macho Agricola 15.29327
## 4 478.53 Macho   Urbano 15.52720
## 5 463.79 Macho Matorral 23.83045
```

```
## 6 412.27 Macho Agricola 46.89978
```

```
# También podemos obtener una muestra aleatoria de un número de filas
muestra11 <- datos %>% sample_n(11)

# Número de filas de muestra11
nrow(muestra11)
```

```
## [1] 11
```

```
muestra11
```

```
##      peso    sex      hab      tri
## 1  416.40 Hembra Matorral 34.97513
## 2  504.73  Macho   Urbano 13.94870
## 3  463.63  Macho   Urbano 28.22643
## 4  455.37  Macho   Urbano 27.69759
## 5  455.81 Hembra Matorral 18.48476
## 6  409.21 Hembra Agricola 32.29201
## 7  416.86 Hembra Matorral 34.17992
## 8  477.46  Macho   Urbano 26.20079
## 9  430.04  Macho Matorral 38.05965
## 10 395.66 Hembra Matorral 32.42623
## 11 451.60  Macho   Urbano 24.28342
```

La ayuda `help()` y preguntar a Google Otra de las ventajas de R con respecto a otros lenguajes de programación es que sus paquetes deben de cumplir una serie de estándares para poder estar en el repositorio "oficial" CRAN. Uno de los requerimientos es que las funciones de los diferentes paquetes deben de estar bien documentadas, es decir, debe existir un *manual* que indique cómo se usa cada una de las funciones del paquete. Ese manual sigue siempre la misma estructura: nombre y descripción de la función, cómo se usa, los argumentos que necesita, el objeto que resulta al aplicar la función (Value), y unos ejemplos de cómo usar la función. Este manual o ayuda puede incluir más apartados, pero los mencionados suelen ser obligatorios. La forma de "llamar a la ayuda" en R es utilizando la función `help()`. Veamos un ejemplo:

```
# Vamos a intentar obtener la media del objeto letras
mean(letras)
```

```
## Warning in mean.default(letras): argument is not numeric or logical: returning
## NA
```

```
## [1] NA
```

```
# No entendemos por qué obtenemos este error... vamos a consultar la ayuda para
# entender qué necesita la función mean()
help(mean)

# Como vemos en el apartado Arguments, a la función mean() sólo se le pueden
# proporcionar objetos que sean de la clase numeric, logical, vectors, date,
# date-time y time_interval.
# Dado que la clase de letras es character, ahora entendemos por qué no podemos
# utilizar mean con letras.
```

Quizás la lección más importante es que normalmente ningún usuario habitual ha asistido a ningún curso especializado de R: la mayoría aprende a base de prueba/error y muchas horas delante del ordendaor con `ERROR` o `WARNINGS` en nuestra pantalla. Por eso es esencial no perder la paciencia y preguntar siempre a Google antes de preguntar al compañero que sabe (siemrpe hay alguien que ha tenido la misma duda antes que nosotros y ha dejado un comentario en un foro o blog). De esta forma interiorizaremos mucho mejor el funcionamiento de este lenguaje y aprenderemos mucho más rápido!

Caso práctico 1

Los fungicidas triazoles son compuestos químicos que se aplican habitualmente en semillas de cultivos para prevenir el crecimiento de hongos patógenos de plantas. Sin embargo, cuando las semillas son consumidas por la fauna silvestre, estos compuestos pueden producir efectos crónicos perjudiciales en su sauld y desarrollo. Queremos estudiar el efecto de los fungicidas triazoles sobre la condición corporal (peso) en perdices rojas (*Alectoris rufa*). Para ello se han capturado un total de 300 perdices en tres hábitats diferentes (semiurbano, agrícola y monte matorralizado) a las que se les ha sexado y extraido muestras de heces para obtener la concentración de fungicidas triazoles (ngramos de compuesto/gramo de heces). La base de datos obtenida está almacenada en este enlace. A continuación iremos realizando algunos test estadísticos vistos en teoría para responder a nuestra pregunta.

```
# Guardamos el archivo en un obejto que denominaremos "datos"
datos <- read.csv("https://raw.githubusercontent.com/jabiologo/web/master/tutorials/estadisticaBasica_f

# Podemos utilizar la función str() para ver la estructura interna del data.frame
str(datos)
```

```
## 'data.frame':    300 obs. of  4 variables:
##  $ peso: num  419 465 434 509 515 ...
##  $ sex : chr  "Macho" "Hembra" "Hembra" "Macho" ...
##  $ hab : chr  "Agricola" "Urbano" "Urbano" "Agricola" ...
##  $ tri : num  47.9 13 18.1 20.9 15.3 ...
```

Vuestro turno

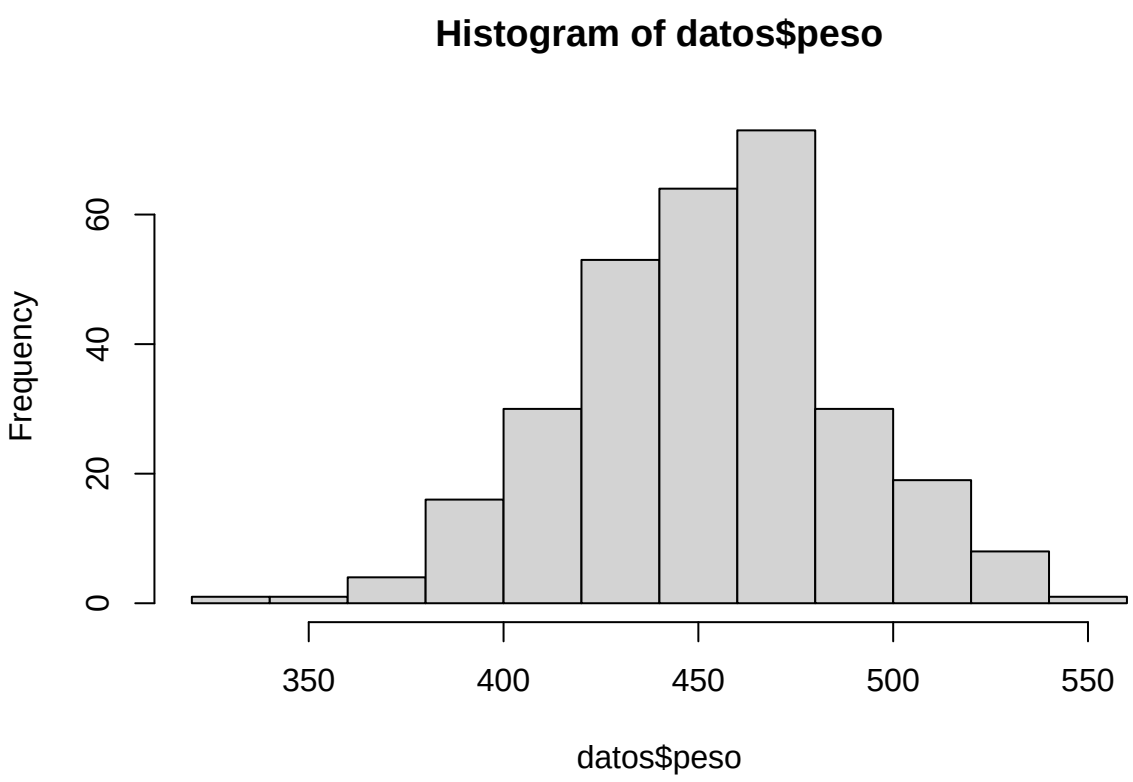
- ¿Cuál es nuestra variable dependiente/respuesta?
- ¿Cuál o cuáles podrían ser nuestras variables independientes o predictores?
- ¿De qué tipo son nuestras variables? Contínuas, categóricas...
- Si hay variables categóricas (factores), ¿cuántos niveles tienen?
- ¿Qué tamaño muestral (n) tengo?
- ¿Están mis datos balanceados?
- ¿Qué distribución sigue mi variable respuesta? ¿y las variables predictoras?

Resolución del caso práctico 1

```
# Primeros elementos de la base de datos
head(datos)
```

```
##     peso    sex      hab      tri
## 1 419.35  Macho Agricola 47.87247
## 2 465.11 Hembra   Urbano 13.03228
## 3 434.17 Hembra   Urbano 18.05707
## 4 508.56  Macho Agricola 20.87707
## 5 514.75  Macho Agricola 15.29327
## 6 478.53  Macho   Urbano 15.52720
```

```
# Histograma de nuestra variable dependiente (peso)
hist(datos$peso)
```




```
# Comprobamos la normalidad de nuestra variable
shapiro.test(datos$peso)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos$peso
## W = 0.9966, p-value = 0.7764
```

```
# ¿Qué significa aquí un p-valor > 0.05?
# ¿Cuáles son aquí nuestras hipótesis nula y alternativa?
# Cómo trabaja la función shapiro.test()?
help("shapiro.test")
```

Recordamos que en el test de Shapiro-Wilk la hipótesis nula es que no existen diferencias entre la distribución de nuestros datos y una distribución normal teórica. Por tanto, si obtuviéramos un P-valor en este test menor de 0.05, indicaría que nuestros datos **no** siguen una distribución normal. Combiene estar alerta en este test, pues siempre estamos acostumbrados a *"buscar P-valores menores de 0.05"*... pero en este caso, lo que *"nos gustaría"* es que el P-valor fuese alto: esto significaría que no hay diferencias entre la distribución de nuestros datos y una distribución Normal teórica. A continuación realizaremos una comparación entre las medias del peso de los macho y de las hembras. Utilizaremos primero el test paramétrico y posteriormente el no paramétrico.

```
# Diferencia entre medias
# ¿Quién pesa más, los machos o las hembras? Filtramos los datos usando tidyverse
datosHe <- datos %>% filter(sex == "Hembra")
datosMa <- datos %>% filter(sex == "Macho")
```

```r
# Recordamos:
# H0 (hipótesis nula): no hay diferencias significativas entre gurpos
# H1 (hipótesis alternativa): hay diferencias significativas entre gurpos

# Paramétrico: T-Student
t.test(datosHe$peso, y = datosMa$peso)
```

```
##
##  Welch Two Sample t-test
##
## data:  datosHe$peso and datosMa$peso
## t = -7.5131, df = 294.22, p-value = 6.974e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -35.46227 -20.74003
## sample estimates:
## mean of x mean of y
##  438.7512  466.8524
```

```r
# No paramétrico: U de Mann-Whitney
wilcox.test(datosHe$peso, y = datosMa$peso)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  datosHe$peso and datosMa$peso
## W = 6161, p-value = 1.276e-11
## alternative hypothesis: true location shift is not equal to 0
```

En este caso vemos que sí que existen diferencias significativas entre el peso de los machos y de las hembras, puesto que en el T-test obtenemos un P-valor muy bajo, `6.974e-13`. Además, el estdístico `t` nos da algo de información adicional... ¿Quién pesa más, los machos o las hembras? ¿Qué le ocurre al estadístico `t` si cambiamos el orden de esta forma: `t.test(datosMa$peso, y = datosHe$peso)`? A continuación realizaremos una comparación entre las medias del peso de de todos los individuos entre los diferentes hábitats. Al igual que en la ocasión anterior, utilizaremos primero el test paramétrico y posteriormente el no paramétrico.

```r
# Diferencia entre medias
# ¿Dónde pesan más? Hábitats urbanos, agrícolas o matorral

# Paramétrico: ANOVA
anova <- aov(peso ~ hab, data = datos)
summary(anova)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## hab           2   9016    4508   3.697 0.0259 *
## Residuals   297 362091    1219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Tukey PostHoc
TukeyHSD(anova, conf.level=.95)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = peso ~ hab, data = datos)
```

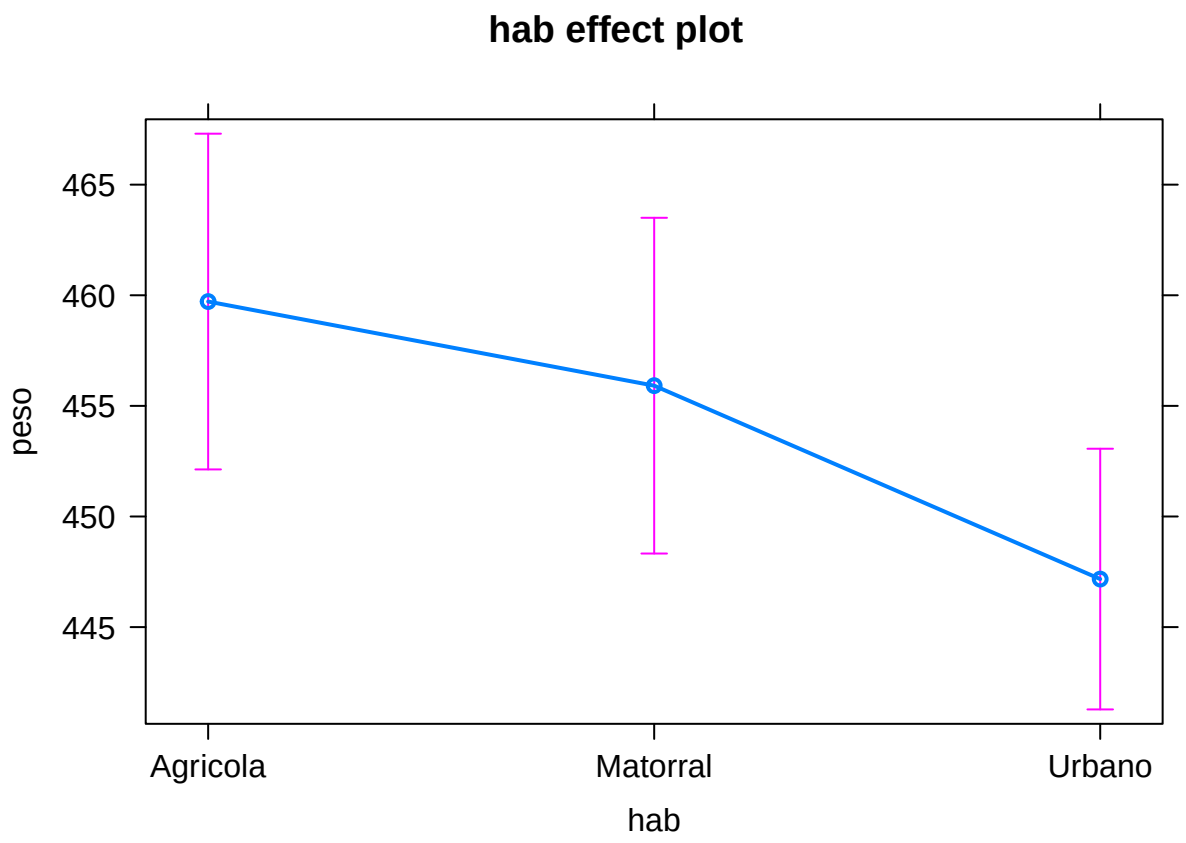
```
##
## $hab
##                          diff       lwr       upr     p adj
## Matorral-Agricola  -3.802317 -16.64709  9.042460 0.7652580
## Urbano-Agricola   -12.544973 -24.04425 -1.045700 0.0286637
## Urbano-Matorral    -8.742656 -20.24193  2.756617 0.1743912
```

```
# Plots (gráficos)
library(effects)
```

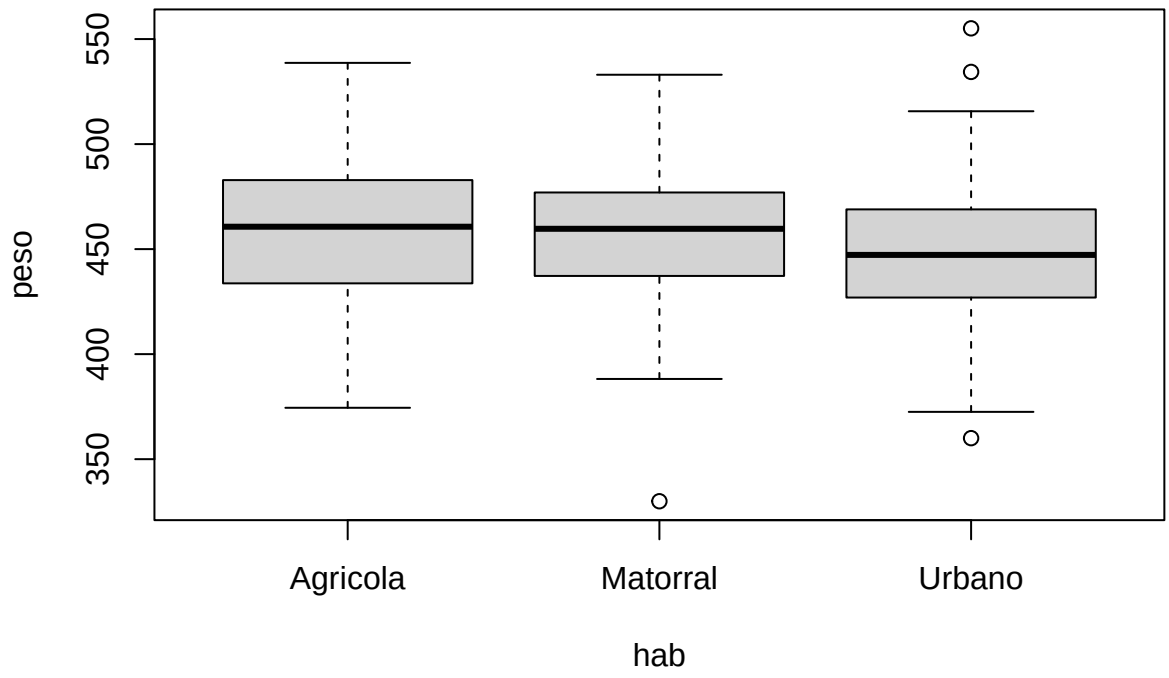
```
## Cargando paquete requerido: carData
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
plot(allEffects(anova))
```




```
boxplot(peso ~ hab, data = datos)
```




```
# No paramétrico: Kruskal-Wallis
kruskal.test(peso ~ hab, data = datos)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  peso by hab
## Kruskal-Wallis chi-squared = 7.1803, df = 2, p-value = 0.02759
```

Merece la pena recordar que en un test ANOVA, el P-valor tan solo nos indica que "existen diferencias significativas entre al menos dos de nuestros grupos", pero no nos informa entre qué grupos están exactamente esas diferencias. Para ello, necesitamos realizar el `TukeyHSD` test *post hoc*. ¿Conoces algún otro tipo de test *post hoc* que nos de la misma información? Finalmente vamos a explorar como varía el peso de nuestros individuos en función de la concentración de triazoles. Para ello utilizaremos primero unas correlaciones (paramétrica primero y no paramétrica después) y posteriormente una regresión lineal.

```
# ¿Cómo afecta la concentración de fungidida al peso?
# Correlación entre dos variables continuas

# Paramétrico: Correlación de Pearson
cor.test(datos$tri, datos$peso, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  datos$tri and datos$peso
## t = -23.23, df = 298, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
##  -0.8395769 -0.7583332
## sample estimates:
##        cor
## -0.8026473
```

```
# No paramétrico: Correlación de Spearman
cor.test(datos$tri, datos$peso, method = "spearman")
```

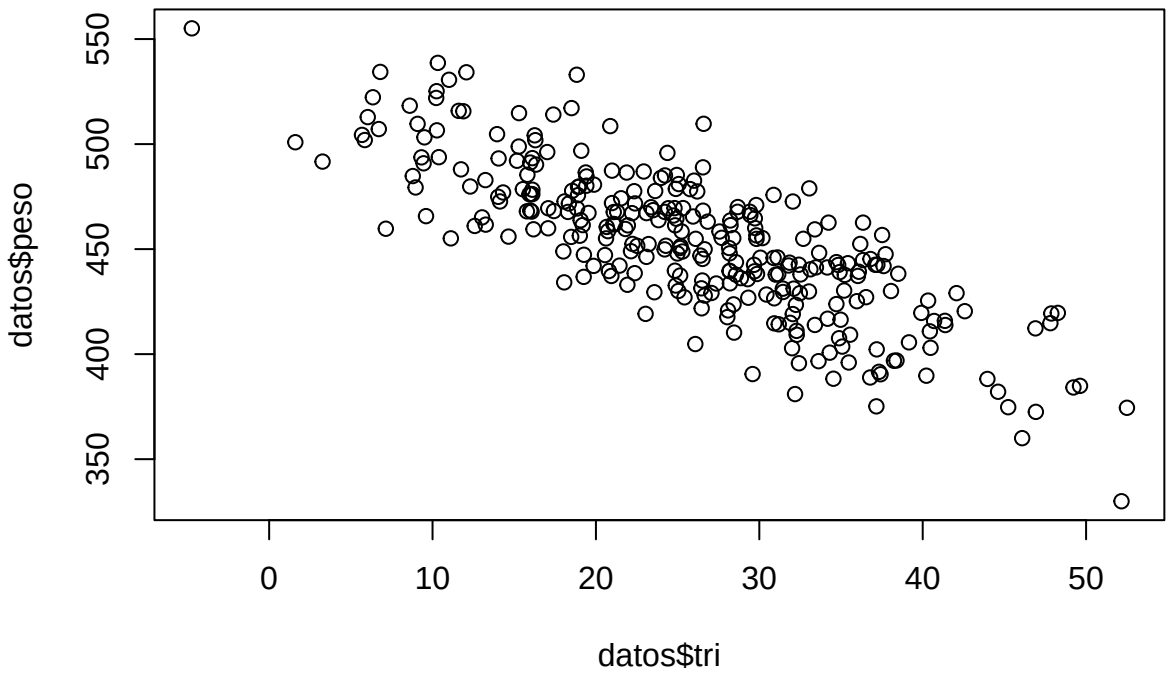
```
## Warning in cor.test.default(datos$tri, datos$peso, method = "spearman"): Cannot
## compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  datos$tri and datos$peso
## S = 8050549, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## -0.7890307
```

```
# Regresión
regresion <- lm(peso ~ tri, data = datos)
summary(regresion)
```

```
##
## Call:
## lm(formula = peso ~ tri, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.822 -15.102  -0.568  15.030  60.303
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 526.1267     3.3748  155.90   <2e-16 ***
## tri          -2.8370     0.1221  -23.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.05 on 298 degrees of freedom
## Multiple R-squared:  0.6442, Adjusted R-squared:  0.643
## F-statistic: 539.6 on 1 and 298 DF,  p-value: < 2.2e-16
```

```
# Gráfico de dispersión (scatter-plot)
plot(datos$tri, datos$peso)
```



¿Qué significa que el coeficiente de correlación de Pearson sea negativo (`-0.8026473`)? ¿Qué relación existe entre este coeficiente y el `Multiple R-squared` obtenido a en la regresión? Es importante recordar que esta regresión nos permite construir la fórmula de la recta con la cual podríamos teóricamente predecir los valores de nuestra variable respuesta en función de nuestra variale predictora.

Finalmente, como se ha comentado en clase, es importante que sepamos que existe un modelo lineal subyacente a cada uno de estos test. Para más información, se recomienda la lectura de este fantástico post de Jonas Kristoffer Lindeløv que queda resumido en la siguiete imagen:

A modo de ejemplo, repetiremos los análisis anteriores utilizando el *framework* de los modelos lineales:

```
# Test paramétricos como modelos lineales

# ¿Quién pesa más, los machos o las hembras?
m1 <- lm(peso ~ sex, data = datos)
summary(m1)
```

```
##
## Call:
## lm(formula = peso ~ sex, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -108.781  -22.150    0.758   21.128   88.248
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  438.751      2.659 164.989  < 2e-16 ***
```

```
## sexMacho      28.101      3.736   7.522 6.41e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.35 on 298 degrees of freedom
## Multiple R-squared:  0.1596, Adjusted R-squared:  0.1567
## F-statistic: 56.58 on 1 and 298 DF,  p-value: 6.411e-13
```

```
# ¿Dónde pesan más? Ambientes urbanos, agrícolas o matorral
m2 <- lm(peso ~ hab, data = datos)
summary(m2)
```

```
##
## Call:
## lm(formula = peso ~ hab, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.943  -21.473    1.327   22.284  107.929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  459.715      3.856 119.224   <2e-16 ***
## habMatorral   -3.802      5.453  -0.697   0.4862
## habUrbano    -12.545      4.882  -2.570   0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.92 on 297 degrees of freedom
## Multiple R-squared:  0.02429,    Adjusted R-squared:  0.01772
## F-statistic: 3.697 on 2 and 297 DF,  p-value: 0.02594
```

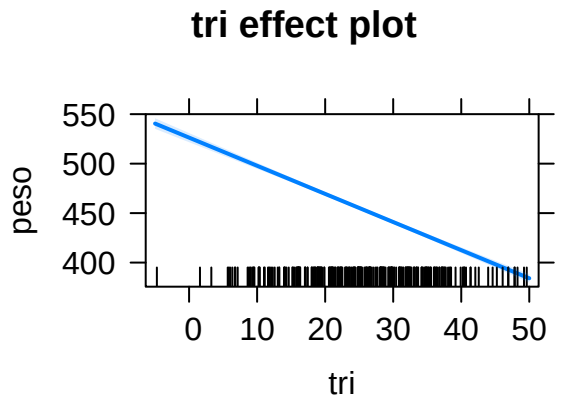
```
# ¿Cómo afecta la concentración de fungidida al peso?
m3 <- lm(peso ~ tri, data = datos)
summary(m3)
```

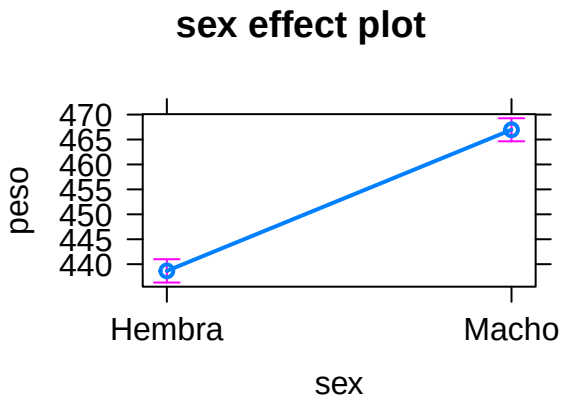
```
##
## Call:
## lm(formula = peso ~ tri, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.822 -15.102  -0.568  15.030  60.303
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 526.1267     3.3748  155.90   <2e-16 ***
## tri          -2.8370     0.1221  -23.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.05 on 298 degrees of freedom
## Multiple R-squared:  0.6442, Adjusted R-squared:  0.643
## F-statistic: 539.6 on 1 and 298 DF,  p-value: < 2.2e-16
```

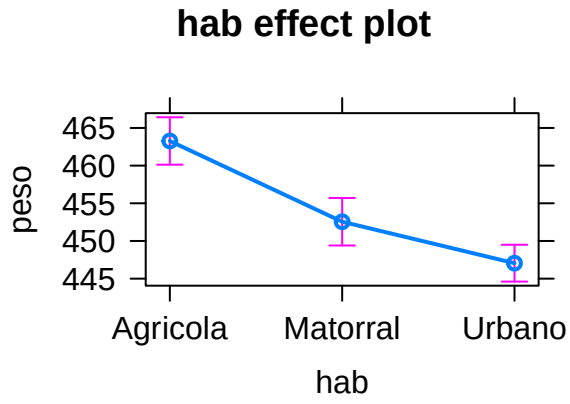
```r
# ¿Cómo afectan la concentración de fungicida, el sexo y el habitat al peso?
m4 <- lm(peso ~ tri + sex + hab, data = datos)
summary(m4)
```

```
##
## Call:
## lm(formula = peso ~ tri + sex + hab, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.866  -9.230  -0.499   9.582  46.738
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 522.22294    2.87952 181.358  < 2e-16 ***
## tri          -2.84291    0.08425 -33.742  < 2e-16 ***
## sexMacho     28.29805    1.67318  16.913  < 2e-16 ***
## habMatorral -10.71749    2.26664  -4.728 3.51e-06 ***
## habUrbano   -16.21739    2.02476  -8.010 2.69e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.45 on 295 degrees of freedom
## Multiple R-squared:  0.8339, Adjusted R-squared:  0.8317
## F-statistic: 370.3 on 4 and 295 DF,  p-value: < 2.2e-16
```

```r
plot(allEffects(m4))
```







Caso práctico 2

Para alcanzar un correcto desarrollo y crecimiento durante las primeras etapas de la vida, los organismos están sujetos a multitud de factores externos que pueden afectarles negativamente. Uno de ellos son las enfermedades transmitidas por patógenos. Queremos estudiar el efecto que las enfermedades producen sobre el crecimiento de pollos de estornino negro (*Sturnus unicolor*) en condiciones naturales. Para ello, se han creado dos grupos experimentales sometidos a un tratamiento: pollos "enfermos", a los cuales se le ha inyectado un antígeno (LPS) y pollos "sanos" (sustancia control, PBS). A estos pollos se les ha medido su masa antes y después del tratamiento para ver el efecto de este sobre su crecimiento. A continuación, procederemos a realizar un análisis detallado con diferentes test estadísticos para responder a nuestra pregunta.

Resolución del caso práctico 2

Los datos para realizar este caso se pueden encontrar en el Campus Virtual de la asignatura.

```
##Procesamiento de datos ####
#Cargamos nuestra base de datos
install.packages("readxl")
library(readxl)
LPS<-read_excel("C:/PONGO_AQUÍ_MI_RUTA/crecimiento_practico.xlsx")

#Miramos que nuestras variables estén en la clase que deben
str(LPS)

#Aquellas con clase numéricas o de caracter, las convertimos a factor
```

```r
LPS$Time<-as.factor(LPS$Time)
LPS$Treatment<-as.factor(LPS$Treatment)
LPS$ID<-as.factor(LPS$ID)
LPS$Sex<-as.factor(LPS$Sex)

#Miro por encima un resumen de nuestras variables (para las continuas te da meadia, etc)
summary(LPS)  #hay NAs (celdas sin datos)

#Si queremos ver un resumen (media, desviación estandar, mediana, máximos, etc) de una variable por gru
install.packages("psych")
library(psych)
describeBy(LPS$mass, LPS$Treatment)

#ojo que en este resumen tenemos datos de masa incluyendo el momento de medición previo y posterior al
#si quisieramos un resumen numerico de la masa, por grupos de tratamiento en el momento post
install.packages("dplyr")
library(dplyr)
LPSPOST<-filter(LPS, Time=="Pre-treatment") #creamos una base de datos filtrando solo para medidas post
describeBy(LPSPOST$mass, LPSPOST$Treatment)



#cambio el orden del factor Time para que el pre sea sobre el que me compara
LPS$Time <- factor(LPS$Time,
                   levels = c("Pre-treatment", "Post-treatment"))

#cambio el orden del factor Treatment para que el control(PBS) sea sobre el que me compara
LPS$Treatment <- factor(LPS$Treatment,
                  levels = c("PBS", "LPS"))



LPS2<-na.omit(LPS)  #eliminamos aquellos pollos para los que no hay medidas en alguno de los momentos


##Exploramos nuestra variable a analizar (masa) ####

#normalidad
shapiro.test(LPS2$mass) #si vemos el test de shapiro, dice que estamos ante una variable NO normal (p n

hist(LPS2$mass) #Sin embargo al graficarlo tampoco vemos una distribución tan rara


#homogeneidad de varianza
install.packages("car")
library(car)
leveneTest(y=LPS2$mass, group=LPS2$Treatment, center="median") #las varianzas no son iguales
#OJO! no son iguales seguramente porque en nuestra variable masa tenemos medidas pre y post y cada una
#tiene una varianza determinada

install.packages("ggplot2")
library(ggplot2)
ggplot(data = LPS2) +
```

```r
  geom_point(aes(x = Treatment, y = mass, colour = Treatment), position=position_jitterdodge()) +
  theme_classic() + theme(legend.position = "none")



#Vemos la normalidad de nuestros datos antes y despues del tratamiento
#Masa antes del tratamiento
LPS2PRE<-filter(LPS2, Time=="Pre-treatment") #filtramos datos pre

shapiro.test(LPS2PRE$mass) #si vemos el test de shapiro, dice que estamos ante una variable normal (p n

hist(LPS2PRE$mass) #en efecto vemos al graficarlo que el peso sigue una distribución normal


leveneTest(y=LPS2PRE$mass, group=LPS2PRE$Treatment, center="median") #las varianzas son iguales

ggplot(data = LPS2PRE) +
  geom_point(aes(x = Treatment, y = mass, colour = Treatment), position=position_jitterdodge()) +
  theme_classic() + theme(legend.position = "none")

#Masa Tras el tratamiento
LPS2POST<-filter(LPS2, Time=="Post-treatment")

shapiro.test(LPS2POST$mass) #si vemos el test de shapiro, dice que estamos ante una variable normal (p

hist(LPS2POST$mass) #en efecto vemos al graficarlo que el peso sigue una distribución normal



leveneTest(y=LPS2POST$mass, group=LPS2POST$Treatment, center="median") #las varianzas son iguales

ggplot(data = LPS2POST) +
  geom_point(aes(x = Treatment, y = mass, colour = Treatment), position=position_jitterdodge()) +
  theme_classic() + theme(legend.position = "none")

##Análisis ####
#Recordamos
# HO (hipótesis nula): no hay diferencias significativas entre gurpos
# H1 (hipótesis alternativa): hay diferencias significativas entre gurpos

###Diferencias iniciales de masa ####

#Hemos cometido algún sesgo en la asignación de tratamientos? Hay diferencias iniciales que puedan dete
#afectarme a mis resultados tras el tratamiento?

#La masa antes del tratamiento es una variable normal con varianzas iguales --> TEST PARAMÉTRICO
#Como quiero comparar medias de una variable cuantitativa (masa) de dos grupos categóricos (LPS Y Contr

#Test paramétrico t de student
t.test(mass ~ Treatment, data = LPS2PRE, alternative="two.sided",   #esta función es diferente a la del
       var.equal=TRUE, conf.level=0.95)
#No existen diferencias iniciales de masa
```

```r
#Vamos a nuestra pregunta principal
###Existen diferencias entre LPS (enfermos) y controles (sanos) tras el desafío inmune? ####

#Al igual que la masa pre tratamiento, la masa en el post es una variable normal
#Test paraétrico. t de student
t.test(mass ~ Treatment, data = LPS2POST, alternative="two.sided",
       var.equal=TRUE, conf.level=0.95)

#No hay diferencias en masa entre controles y experimentales tras el tratamiento

#Pero demosle la vuelta a la pregunta

###Existen diferencias en el crecimiento de los pollos acorde al tratamiento? ####
#es decir, queremos ver si, en lugar de diferencias antes y después, hay diferencias en el crecimiento,
#el cambio de peso entre ambas edades?
#Para ello echamos mano de la variable Time, que agrupa medidas hechas antes y después del tratamiento

install.packages("tidyverse")
library(tidyverse)
install.packages("ggpubr")
library(ggpubr)
install.packages("rstatix")
library(rstatix)

#Vamos a buscar en este caso una interacción entre tratamiento y tiempo de la medida (pre, post)
#Por tanto, si la masa es una variable normal y queremos ver si se predice por la interacción de
#dos variables categóricas (Treatment y Time) --> ANOVA de dos vias

#ANOVA DE DOS VIAS
anova<-aov(mass ~ Treatment * Time, data=LPS2)
summary(anova)

#Nos fijamos en el P valor de la interacción únicamente
#No sale significativo

#OJO! En realidad hemos cometido un ERROR!
#No le hemos dicho al modelo que un mismo individuo se ha medido antes y después del tratamiento
#Osea, que las medidas de masa no son independientes, sino que dependen una de otra porque son del mism

#ANOVA DE DOS VIAS CON MEDIDAS REPETIDAS
anovarep<-aov(mass ~ Treatment*Time + Error(ID/Time), data=LPS2) #introducimos el argumento Error dicie
summary(anovarep)
#La interacción ahora SI es significativa

#Graficamos la interacción
ggplot(data = LPS2, aes(x = Time, y = mass, colour = Treatment,
                        group = Treatment)) +
  stat_summary(fun = mean, geom = "point", size=3) +
  stat_summary(fun = mean, geom = "line", size=1) +
  labs(y  =  'Mass', x="") +
  #geom_point(mapping = aes(x = Time, y = mass, color=Treatment), size=2, alpha=0.2, stroke=1,
   #          position=position_jitterdodge()) +
  theme_classic() +
```

```
  theme(plot.margin = margin(b= 25, l=25),
        axis.title.x = element_text(vjust = -1, colour="black", family="sans", size = 22),
        axis.title.y = element_text(vjust = 4, size = 22, family="sans"),
        axis.text.y = element_text(size = 20, family="sans"),
        axis.text.x = element_text(size = 20, family="sans"),
        legend.text = element_text(colour="black", family="sans", size = 20),
        #legend.justification=c(0.95,0.1), legend.position=c(0.95,0.1),
        legend.background = element_rect(fill = "white", colour=1),
        legend.position = "top",
        legend.title = element_blank())

#Mientras qu elos LPS crecen poquito o nada, los PBS crecen mucho más

#CONCLUSIÓN: Estar enfermos te hace crecer menos

###OTRA APROXIMACIÓN: MODELOS LINEALES MIXTOS permiten la incorporación de diferentes variables indepen
#regresiones multiples y de incorporar efectos aleatorios (muestras dependientes como un anova de medid

install.packages("lmerTest")
library(lmerTest)
install.packages("lme4")
library(lme4)
install.packages("Matrix")
library(Matrix)

m1<-lmer(mass ~ Treatment*Time + (1|ID), data=LPS2)  #para decir al modelo que son muestras dependiente

#Exploramos la normalidad de los residuos del modelo
plot(m1) #buscamos una nube homogenea


#sacamos los resultados
summary(m1)

#ploteamos los efectos
install.packages("effects")
library(effects)
plot(allEffects(m1))




#Podemos incluir diferentes covariables u otros factores que consideremos que ayudan a explicar la vari
#de la masa de los pollos

m1<-lmer(mass ~ Treatment*Time + Date + Sex + (1|ID), data=LPS2)

#Exploramos la normalidad de los residuos del modelo
plot(m1)


#sacamos los resultados
summary(m1)
```

```
plot(allEffects(m1))
```